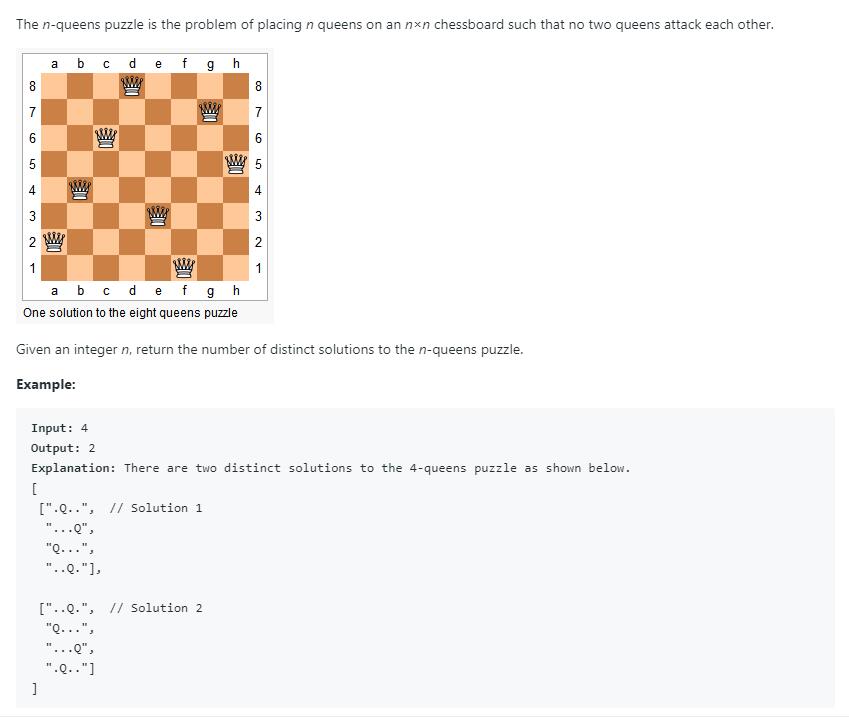
<https://leetcode.wang/leetCode-52-N-QueensII.html>

**题目描述（困难难度）**



和[上一题](https://leetcode.windliang.cc/leetCode-51-N-Queens.html)一样，只不过这次不需要返回所有结果，只需要返回有多少个解就可以。

**解法一**

我们直接把上道题的 ans 的 size 返回就可以了，此外 currentQueen.size ( ) == n 的时候，也不用去生成一个解了，直接加一个数字占位。

public int totalNQueens(int n) {

List<Integer> ans = new ArrayList<>();

backtrack(new ArrayList<Integer>(), ans, n);

return ans.size();

}

private void backtrack(List<Integer> currentQueen, List<Integer> ans, int n) {

if (currentQueen.size() == n) {

ans.add(1);

return;

}

for (int col = 0; col < n; col++) {

if (!currentQueen.contains(col)) {

if (isDiagonalAttack(currentQueen, col)) {

continue;

}

currentQueen.add(col);

backtrack(currentQueen, ans, n);

currentQueen.remove(currentQueen.size() - 1);

}

}

}

private boolean isDiagonalAttack(List<Integer> currentQueen, int i) {

int current\_row = currentQueen.size();

int current\_col = i;

for (int row = 0; row < currentQueen.size(); row++) {

if (Math.abs(current\_row - row) == Math.abs(current\_col - currentQueen.get(row))) {

return true;

}

}

return false;

}

Copy

时间复杂度：

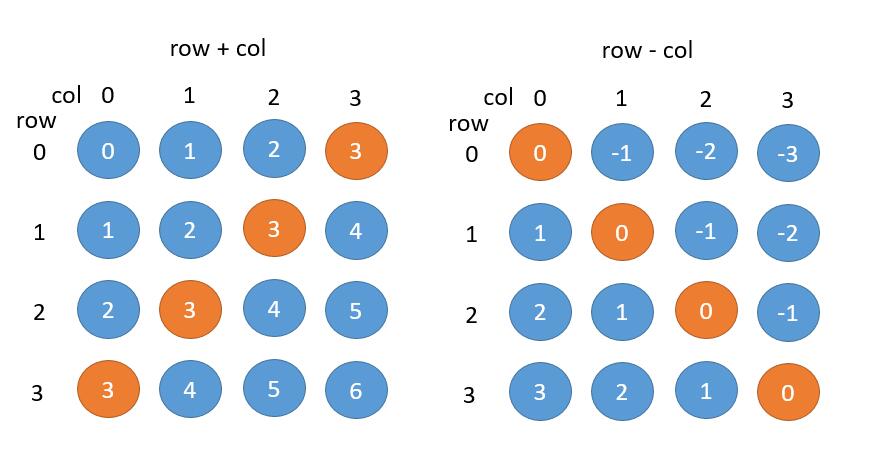
空间复杂度：

**解法二**

参考[这里](https://leetcode.com/problems/n-queens-ii/discuss/20048/Easiest-Java-Solution-(1ms-98.22))。

既然不用返回所有解，那么我们就不需要 currentQueen 来保存当前已加入皇后的位置。只需要一个 bool 型数组，来标记列是否被占有就可以了。

由于没有了 currentQueen，所有不能再用之前 isDiagonalAttack 判断对角线冲突的方法了。我们可以观察下，对角线元素的情况。



可以发现对于同一条副对角线，row + col 的值是相等的。

对于同一条主对角线，row - col 的值是相等的。

我们同样可以用一个 bool 型数组，来保存当前对角线是否有元素，把它们相加相减的值作为下标。

对于 row - col ，由于出现了负数，所以可以加 1 个 n，由 [ - 3, 3 ] 转换为 [ 1 , 7 ] 。

public int totalNQueens(int n) {

List<Integer> ans = new ArrayList<>();

boolean[] cols = new boolean[n]; // 列

boolean[] d1 = new boolean[2 \* n]; // 主对角线

boolean[] d2 = new boolean[2 \* n]; // 副对角线

return backtrack(0, cols, d1, d2, n, 0);

}

private int backtrack(int row, boolean[] cols, boolean[] d1, boolean[] d2, int n, int count) {

if (row == n) {

count++;

} else {

for (int col = 0; col < n; col++) {

int id1 = row - col + n; //主对角线加 n

int id2 = row + col;

if (cols[col] || d1[id1] || d2[id2])

continue;

cols[col] = true;

d1[id1] = true;

d2[id2] = true;

count = backtrack(row + 1, cols, d1, d2, n, count);

cols[col] = false;

d1[id1] = false;

d2[id2] = false;

}

}

return count;

}

Copy

时间复杂度：

空间复杂度：

**总**

和上一题相比，通过三个 bool 型数组来标记是否占有，不存储具体的位置，从而解决了这道题。